

⑫ 公開特許公報 (A)

昭63-12096

⑪ Int. Cl.

識別記号

庁内整理番号

⑬ 公開 昭和63年(1988)1月19日

G 08 G 1/12
G 01 C 21/006821-5H
Z-6666-2F
N-6666-2F
H-8302-2C
Z-7443-3D// G 09 B 29/10
B 60 R 11/02

審査請求 未請求 発明の数 1 (全20頁)

⑭ 発明の名称 車両の現在地認識方法

⑮ 特 願 昭61-156883

⑯ 出 願 昭61(1986)7月2日

⑰ 発 明 者 安 藤 齊 埼玉県所沢市花園4丁目2610番地 バイオニア株式会社所
沢工場内⑱ 発 明 者 柏 崎 隆 埼玉県所沢市花園4丁目2610番地 バイオニア株式会社所
沢工場内⑲ 発 明 者 細 井 雅 幸 埼玉県所沢市花園4丁目2610番地 バイオニア株式会社所
沢工場内

⑳ 出 願 人 バイオニア株式会社 東京都目黒区目黒1丁目4番1号

㉑ 代 理 人 弁理士 藤村 元彦

明 細 書

1. 発明の名称

車両の現在地認識方法

2. 特許請求の範囲

地図の道路上の各位置を数値化して地図データとして記憶しておき、走行距離センサの出力に基づいて一定距離だけ走行したことを検出する毎に、前記地図データに基づいて前記道路上の前回検出位置から前記一定距離だけ離れた前記道路上の位置を検出し、この検出位置を現在地として認識することを特徴とする車両の現在地認識方法。

3. 発明の詳細な説明

技術分野

本発明は、車載ナビゲーション装置における車両の現在地認識方法に関するものである。

背景技術

近年、地図情報をメモリに記憶しておき、その地図情報をメモリから読み出して車両の現在地とともに認識装置に認識させることにより、車両を

所定の目的地に誘導する車載ナビゲーション装置が研究、開発されている。

かかるナビゲーション装置では、車両に搭載された走行距離センサや方位センサ等の出力に基づいて車両の走行距離や方位等を検出し、これに基づいて時々刻々と変化する車両の現在地を推測することにより、ディスプレイに認識されている地図上への現在地の認識が行なわれる。

この場合、現在地は常に地図の道路上に認識されるのが好ましいのであるが、センサの精度、計算誤差、地図精度等により、特に走行距離が長くなるにつれて、認識上、現在地が道路から外れてしまうことになる。

発明の概要

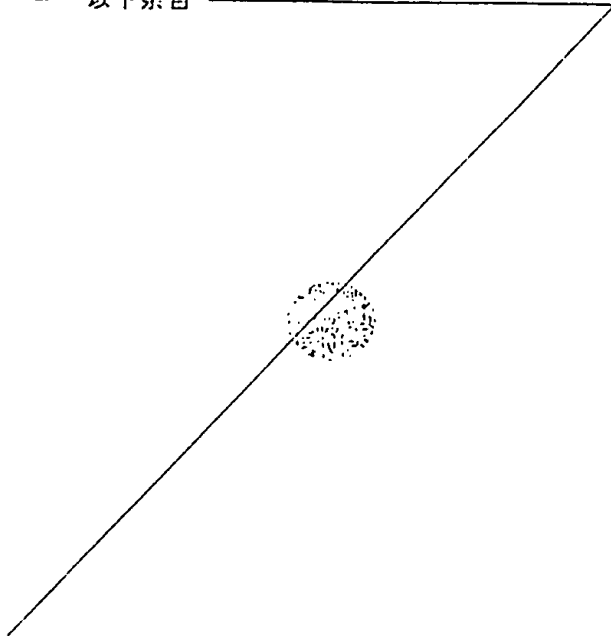
本発明は、上述した点に鑑みなされたもので、車両の現在地を常に正確に認識し得る車両の現在地認識方法を提供することを目的とする。

本発明による車両の現在地認識方法は、地図の道路上の各位置を数値化して地図データとして記憶しておき、走行距離センサの出力に基づいて一

正 地 図

定距離だけ走行したことを検出する毎に、地図データに基づいて道路上の前回検出位置から前記一定距離だけ離れた道路上の位置を検出し、この検出位置を現在地として認識することを特徴としている。

——以下余白



以下、本発明の実施例を図に基づいて説明する。

第1図は、本発明に係る車載ナビゲーション装置の構成を示すブロック図である。同図において、1は地磁気に基づいて車両の方位データを出力するための地磁気センサ、2は車両の角速度を検出するための角速度センサ、3は車両の移動距離を検出するための走行距離センサ、4は緯度及び経度情報等から車両の現在地を検出するためのGPS(Global Positioning System)装置であり、これら各センサ(装置)の出力はシステムコントローラ5に供給される。

システムコントローラ5は、各センサ(装置)1~4の出力を入力としA/D(アナログ/デジタル)変換等を行なうインターフェース6と、種々の画像データ処理を行なうとともにインターフェース6から順次送られてくる各センサ(装置)1~4の出力データに基づいて車両の移動量等を演算するCPU(中央処理回路)7と、このCPU7の各種の処理プログラムやその他必要な情報

が予め書き込まれたROM(リード・オンリ・メモリ)8と、プログラムを実行する上で必要な情報の書き込み及び読出しが行なわれるRAM(ランダム・アクセス・メモリ)9と、いわゆるCD-ROM、ICカード等からなり、デジタル化(数値化)された地図情報が記録された記録媒体10と、V-RAM(Video RAM)等からなるグラフィックメモリ11と、CPU7から送られてくる地図等のグラフィックデータをグラフィックメモリ11に描画しかつ画像としてCRT等のディスプレイ12に表示すべく制御するグラフィックコントローラ13とから構成されている。入力装置14はキーボード等からなり、使用者によるキー入力により各種の指令等をシステムコントローラ5に対して発する。

記録媒体10には地図情報が記録されるのであるが、そのデータ構造について以下に説明する。先ず、第2図(A)に示すように、日本全国を例えば16384(=2¹⁴)[m]四方のメッシュに分割し、このときの1つのメッシュをテリトリ

ーと呼ぶ。テリトリ-はテリトリ-No. (Tx, Ty)で識別され、各テリトリ-には例えば図の左下のテリトリ-を基準にテリトリ-No. が付与される。テリトリ-No. は現在地(Crntx, Crnty)より求まる。テリトリ-は本データ構造の中で最も大きな管理単位となる。地図データファイル全体の構成が第2図(B)に示されており、テリトリ-IDファイルには、第2図(C)に示すように、テリトリ-No. (Tx, Ty)のファイルにおける先頭アドレス、テリトリ-の左下の緯度(実数)、テリトリ-の左下の経度(実数)、地磁気の偏角(実数)等のデータが各テリトリ-毎に書き込まれている。

テリトリ-ファイルは本データ構造において最も重要なファイルであり、各種の地図データや地図描画に必要なデータが書き込まれている。第3図(A)において、ナビID及びセクションテーブルがナビゲーションにおける道路及び交差点検索用ファイル、ピクチャーIDが表示管理用ファイル、道路セクションデータから交差点データま

でが実際の地図データである。地図データは、第3図(B)に示すように、階層構造となっており、最下層が川、海、湖等のポリゴンデータ、その上が道路、鉄道等のラインデータ、その上が各種マーク等のキャラクタデータ、その上が地名等の文字データ、そして最上層が交差点データとなっている。最上層の交差点データは後述する交差点引込みのために用いられるデータであり、ディスプレイ上には表示されない。

次に、第4図(A)に示すように、1つのテリトリの中を例えば256分割し、これにより切られる1024(2^4)[m]四方のメッシュをユニットと呼ぶ。このユニットも同様にユニットNo. (Nx, Ny)で管理され、そのNo.

(Nx, Ny)は現在地(Crntx, Crnty)より求まる。ユニットは中間的な管理単位で、地図情報はこの単位で記録され、ユニットが256個集まってテリトリファイルを構成する。地図描写の際はこの単位を基に行なわれるので、描画の基本単位とすることができる。ナビIDファイルに

うファイルがある。本実施例では、地図データの縮尺が例えば2.5万分の1、5万分の1、10万分の1の3種類に設定されており、実際の地図データとしては、最も縮尺の大きい2.5万分の1のものだけを持っている。各縮尺の地図は、第8図～第10図の各図(A)に示すように、エリアに分割され、このエリアはエリアNo. (Anx, Any)で管理される。エリアNo. (Anx, Any)は現在地(Crntx, Crnty)より求まる。縮尺が2.5万分の1の場合、エリアNo. とユニットNo. は同じであり、5万分の1の場合は1つのエリアがユニットファイル4個分となり、10万分の1の場合は1つのエリアがユニット16個分となる。また、各縮尺のピクチャーIDには、第8図～第10図の各図(B)にそれぞれ示すように、その縮尺の地図を表示するのに必要なポリゴン、ライン、キャラクタ、文字データの先頭アドレスとデータサイズが記録されている。

続いて、ポリゴンデータとラインデータについて説明する。ポリゴンデータとラインデータは、

は、第4図(B)に示すように、ユニットNo. (Nx, Ny)のファイルにおけるライン先頭アドレス、交差点先頭アドレス、道路セクション先頭アドレス、交差点先頭アドレス等のデータが各ユニット毎に書き込まれている。

更に、第5図(A)に示すように、1つのユニット内を例えば16分割し、これにより切られる256(2^4)[m]四方のメッシュをセクションと呼ぶ。このセクションも同様にセクションNo. (Sx, Sy)で管理され、そのNo. (Sx, Sy)は現在地(Crntx, Crnty)より求まる。セクションは最も小さい管理単位であり、この範囲内の線分(線分の繋りで道路等が表わされる)や交差点の情報が第5図(B), (C)に示す如くセクションテーブルとして、更に第6図(A), (B)及び第7図(A), (B)に示す如くセクションデータとしてテリトリファイルに登録されている。

また、第3図(A)に示すように、テリトリファイル内には表示管理用のピクチャーIDとい

第11図(A)及び第12図(A)に示すように、始点と終点で表わされる繋りのあるベクトル(線分)で表わされる。ここで、最も縮尺の大きい2.5万分の1の地図データで5万分の1や10万分の1の地図を表現すると、始点・終点間が縮まるのでディスプレイ上で見た限りでは、全ての点を表示しなくても差し支えないことがある。このことを考慮に入れて、ディスプレイ上に表示した場合に、見た目上省略しても差し支えない点の情報を、第11図(B)及び第12図(B)に示すように、予めポリゴン及びラインデータの各間引きビットに入れておく。そして、各縮尺の表示時に間引きビットをチェックして必要に応じて間引きビットに情報が入っている点を除く、いわゆる間引きを行なうことにより、表示する線分(ベクトル)数を減らすことができる。

また、第13図(A)に示すように、1ユニット内に存在する交差点の全てに通し番号(xn, yn)が付されている。ところで、交差点には、直交型、Y字路、5叉路等種々あるが、特に方位の似

た道路が複数入っている交差点では、この交差点を通過したときに、センサの精度、計測誤差、地図精度等により道路の選択を誤り、ディスプレイ上に現在地が表示されている道路と実際に走行している道路とが一致しない状態が生ずる可能性がある。そこで、このような交差点に対しては、第13図(B)に示すように、交差点の難易度を示す難易度データを交差点データ中の難易度ビットにいておく。そして、交差点を通過するときには、この難易度データに基づいた処理を行えば、誤った道路の選択を防止できるのである。その処理に関しては後で説明する。

次に、地図データの表示に関して、グラフィックメモリ11として例えばV-RAMを用いた場合について説明する。表示の構成としては、第14図(A)に示すように、512(ドット)×512(ドット)のV-RAM上で画面を16分割し、それぞれのエリアに独立した1枚の地図を表示するようにする。1エリアは128(ドット)×128(ドット)の1ユニットであり、更に1

6分割することにより1エリアは32(ドット)×32(ドット)の1セクションとなる(第14図(B)、(C)を参照)。実際の車載ディスプレイには、第14図(A)の中央の4画面に相当する256(ドット)×256(ドット)のエリア(太線で囲ったエリア)が表示され、このエリアがV-RAM上を移動することによって車両の現在地の動きを表現する。

次に、CPU7によって実行される基本的な手順を第15図のフローチャートに従って説明する。

CPU7は、先ず最初にプログラムを実行させるためのイニシャライズを行ない(ステップS1)、しかる後車両の現在地がセットされているか否かを判断する(ステップS2)。現在地がセットされていない場合は、現在地セットルーチンの実行(ステップS3)、例えば入力装置14でのキー入力による現在地のセットが行なわれる。次に、走行距離を零にし(ステップS4)、続いて入力装置14からのキー入力があるか否かの判断を行なう(ステップS5)。

キー入力がない場合は、ディスプレイ12上に現在地周辺の地図表示を行なうとともに車両の現在位置及びその方位を例えば車両マークにてこの地図上に表示し、車両が移動したらその移動に伴い地図をスクロールさせ、更に現在グラフィックメモリ11上にある地図データの範囲を車両位置が越えそうなどときには、記録媒体10から必要な地図データを読み出してディスプレイ12上に表示する(ステップS6)。

キー入力があると、その入力データに応じて現在地の再セット(ステップS7)、センサ補正(ステップS8)、目的地セット(ステップS9)及び地図の拡大・縮小(ステップS10)の各ルーチンを実行する。

また、CPU7はタイマーによる割込みにより、第16図に示すように、一定時間間隔で地磁気センサ1及び角速度センサ2の各出力データに基づいて車両の方位を常に計算する処理を行なう(ステップS11、S12)。

CPU7は更に、走行距離センサ3よりデータ

が入力された場合は、走行距離センサによる割込み処理を行なう。この割込み処理では、第17図に示すように、走行距離と方位からの現在地の算出(ステップS13)、右折、左折の判定(ステップS14)、道路への引込み(ステップS15)、交差点引込み(ステップS16)、走行距離による引込み(ステップS17)が実行される。なお、このステップS13～ステップS17における各処理に関しては、後で詳細に説明する。

また、GPS装置4より得られる精度、経度データは、第18図に示すように、GPSデータ受信割込みにより処理され、現在地データとして座標変換される(ステップS18)。

車両の走行距離は走行距離センサ3の出力から求められる。この走行距離センサ3としては、例えば、車のいわゆるスピードメータケーブルの回転数(JIS規格では、637回転/Km)より1回転の距離を積分することにより走行距離を求める構成のものが用いられるが、センサ3の精度により得られる走行距離に誤差が生ずることは避

けられない。また、センサ3の精度だけではなく、地図の精度、タイヤの空気圧の変化、或はスリップ等も走行距離の誤差の要因となる。従って、走行距離の補正を度々行なわないと、正確に距離を求めることができなくなってしまうことになる。このため、走行距離センサ3の出力より得た実際の距離と地図データより得た距離とから距離補正係数 r_s を求め、この補正係数 r_s を用いて距離補正を行なうことにより、走行距離を常に正確に検出できるのである。

また、車両の方位は地磁気センサ1の出力から求められる。この方位検出方法に関しては、本出願人等による特願昭60-282341号明細書等に記載されている。この地磁気センサ1の示す北は磁北であり、地図北ではない。このため、磁北が地図北に対してずれていた場合、第19図に示すように、基準位置から一定距離だけ走行したときに地磁気センサ1の出力から得られる推測現在地 P_1 は実際の現在地 P_2 に対してずれを生じることになる。そのため、地磁気センサ1より求

めた方位を地図方位に変換する作業が必要となる。この変換作業は、第20図に示すように、2次元幾何の座標変換で求まる回転角、即ち方位補正係数 θ_s によって行なわれる。この方位補正係数 θ_s は地域により変化し、更に地磁気センサ1を車体に取り付けたときに生じる取付け誤差によっても変わる。この方位補正係数 θ_s は、第21図に示すように、当該係数を零として位置のわかっている2点間を走行し、慣性航法により求められた現在地と到着点との誤差により求めることができる。この方位補正係数 θ_s を用いて方位補正を行なうことにより、車両の方位を常に正確に検出できるのである。

なお、距離補正係数 r_s 及び方位補正係数 θ_s の算出方法は、本出願人等による特願昭60-282344号明細書等に記載されている。

次に、CPU7によって実行される走行距離センサ3による割込み処理の手順を、第22図のフローチャートに従って説明する。走行距離センサ3の出力データにより、現在地の推測地点が随時

計算されており、現在地認識ルーチンとして、本ルーチンが所定のタイミングで呼び出される。

CPU7は先ず、単位距離 l_0 を走ったか否かを判断する(ステップS20)。ここに、単位距離とは、車両が実際に走行した一定の道程を言い、例えば20[m]に設定されている。そして、一定走行距離毎に本ルーチンが実行され、先ず地図データとの関係即ち、第23図に示す如く最近傍線分 l までの距離 l_1 、その線分 l の地図北となす角度 θ_n 等を求め、更にほぼ等距離に2本以上の線分があるときは、その旨をフラグで示す(ステップS21)。その他、近傍交差点の有無などをここで求めるようにしても良い。続いて、距離 l_1 が予め設定した閾値 l_{th} を超えたか否かを判断する(ステップS22)。超えていなければ、ほぼその線分近傍に現在地があるとして、誤差分 l_1 の修正を行なう(ステップS23)。この誤差分 l_1 は、走行距離センサ3の検出誤差、地図データのデジタイズ誤差等に起因するものである。この修正を行なうのは、次の現在地の認識のため

には、それらの誤差をキャンセルしておく必要があるためである。この後、後述するパターン引込みルーチンに進む。

一方、距離 l_1 が閾値 l_{th} を超えている場合は、次に車両がカーブ(右折又は左折)したか否かを判断する(ステップS24)。カーブの検出方法については、後で別に述べる。カーブしなかった場合、地磁気センサ1の出力データから得られた車両の進行方位 θ と線分 l の角度 θ_n の差を設定基準値 θ_{th} と比較する(ステップS25)。 $|\theta - \theta_n| > \theta_{th}$ ならば、何もせずにパターン引込みルーチンに進む。このケースとしては、例えば、T字路をつき当り方向に進んだり、或は地図データとして記憶されていない道を走っているような場合が考えられる。続いて、近傍にY字路等、より小さい角度をもった難易度の高い交差点があるか否かを判断する(ステップS26)。近傍に例えばY字路がある場合には、現在走っている道路とは別の道路に引き込んでしまう可能性があるため、何もせずにパターン引込みルーチンに進む。

交差点の難易度を示すデータは、地図を数値化する際に予め第13図(B)に示す如く交差点データの難易度ビットに挿入されているので、CPU 7はステップS26でこのビットをチェックすれば良いのである。

以上の2つの条件が当てはまらないときは、センサ等の誤差が生じて道路データから外れつつあると判断し、この場合は、現在地の修正、即ち道路データへの引込みを行なう(ステップS27)。新しい現在地の推測地点Pcpdは、第24図に示すように、センサ出力から求めた前回推測地点Pdpdからの相対関係より演算された現在推定地点Pcpより、最近傍線分しにおろした垂線と交わる点とし、表示等を変更する。距離 $l_{\#}$ 及び現在推定地点Pcpdの座標($X_{\#}$, $Y_{\#}$)はその地点における修正値として、後で述べるパターン引込みルーチンで使用するため記憶される。

ステップS24でカーブしたと判断した場合、交差点引込みルーチンに入る。先ず、前回交差点として認識した地点からの走行距離 l_c を求め、

を新しい認識交差点として記憶(更新)する。これにより、車両の現在地がディスプレイ12上に表示されている地図の道路上から外れた場合に、強制的に地図上の交差点上に車両の現在地をのせる、いわゆる交差点引込みが行なわれるのである。続いて、前回認識した交差点の座標、そこからの修正値の和及び現在地の座標に基づいて距離及び方位の補正係数 r_s , θ_s を更新する(ステップS33)。このように、交差点を認識する毎又は交差点間の距離が長い場合には一定距離 l_0 だけ走行する毎に、距離及び方位の補正係数 r_s , θ_s の更新を行えば、より精度の高い現在地推測が可能となる。

次に、パターン引込みについて説明する。このルーチンは一定距離 l_0 だけ走った時点で実行される。距離 l_0 は、例えば1000[m]という値である。なお、ステップS31で交差点認識が行なわれた場合には、走行距離はリセットされる。一定距離 l_0 だけ走行する間に、最近傍線分までの距離 $l_{\#}$ が、 $n = l_0 / l_0$ [回] 測定される

この走行距離 l_c に対して一定値 a_c を乗じたものを、交差点検出閾値 l_{clh} とする(ステップS28)。一定値 a_c は走行距離センサ3の精度に関連した値で、例えば0.05程度の値とする。地図データとして入っている交差点データに対し、現在地Pcpから各交差点までの距離 l_c を求め(ステップS29)、 $l_c < l_{clh}$ なる交差点が存在するか否かを判断する(ステップS30)。ステップS30では、一定距離範囲(例えば、数百[m]程度)以内か否かの判断も行なう。ここで、交差点が存在しなかった場合には、パターン引込みルーチンに進む。また、近傍交差点が複数あり、かつ交差点までの距離 l_c が同程度で近傍交差点を特定できないと判断(ステップS31)した場合も、パターン引込みルーチンに進む。

近傍交差点が特定された場合、その交差点を新しい現在地推測地点Pcpdとして引込みを行なう(ステップS32)。この際、交差点までの距離 l_c 及び現在地Pcpの座標(X_c , Y_c)は引込み値として記憶される。また、現在地推測地点Pcpd

ことになり、 n 回の誤差修正値 e_i がデータとして記憶されている。更に、1回の測定に対し、前回測定時の誤差修正値 e_{i-1} と今回の誤差修正値 e_i との差を、変化量 c_i ($e_i - e_{i-1}$)として計算しておくものとする。

一定距離 l_0 だけ走ったと判断したら(ステップS34)、変化量 C_i の値のはらつきについて計算を行なう。先ず、平均値 $c_{\#}$ ($= \frac{1}{n} \sum_{i=1}^n c_i$)を計算し(ステップS35)、続いてその偏差 α ($= \frac{1}{n} \sum_{i=1}^n (c_{\#} - c_i)$)を計算する(ステップS36)。そして、この値 α を予め定めた閾値 α_{thh} と比較する(ステップS37)。この値 α は、各センサの検出誤差、走行距離等を考慮して求めたものである。 $\alpha > \alpha_{thh}$ の場合、引込み不可能と判断し、パターン引込みは行なわない。一方、 $\alpha < \alpha_{thh}$ の場合、更に現在引込みが行なわれているか否かを判断し(ステップS38)、引込みが行なわれていない場合、即ち道路データから外れた位置に現在地がある場合、現在地の最近傍線分への引込みを行なう(ステップS39)。更に、

閾値 α_{thh} と同様に定められた閾値 α_{thl} と比較し(ステップS40)、 $\alpha < \alpha_{thl}$ のときには、距離及び方位の補正係数 r_s, θ_s を更新する(ステップS41)。

以上の方法で、一度道路から外れたところを走行した後、他の道路に再引込みを行なうことが可能となる。すなわち、デジタイズされていない道路を走行し、再びデジタイズされた道路を走行すると、一定距離を走った時点でその道路が認識され、精度の良い現在地推測が可能となる。また、一定距離 l_{p0} に対し、より長い距離 l_1 について偏差計算を行ない、距離 l_{p0} を短くして精度を上げ、応答時間を短くすることも可能である。第25図(A)、(B)に、その様子を示す。

以上のようにして、最近傍交差点への引込みや最近傍線分への引込みが行なわれるのであるが、この引込みを行なうためには、現在地に最も近い道路(最近傍線分)や交差点(最近傍交差点)を探し出す作業が必要となる。この最近傍交差点や最近傍線分をサーチする作業は、線分や交差点デ

ータの道が多い、即ちサーチエリアが広いと、時間がかかってしまい、時々刻々と変化する現在地をスムーズに表示できないことになる。ところが、本実施例においては、第2図～第5図に示したデータ構造から明らかなように、現在地からのサーチエリアをできるだけ小さくし、かつそのエリアに入る線分や交差点のデータを管理するデータ(セクションデータ、セクションテーブル)を持たせていることにより、最小単位のセクションをサーチエリアとしてその中から線分や交差点をサーチすることができるので、サーチに要する時間を短縮できるのである。以下、CPU7によって実行される現在地から最近傍線分と最近傍交差点をサーチする手順を、第26図のフローチャートに従って説明する。

CPU7は先ず、現在地($Crntx, Crnty$)からテリトリリーNo. (Tx, Ty)、ユニットNo. (Nx, Ny)、セクションNo. (Sx, Sy)をそれぞれ求める(ステップS50～S52)。これは、各エリアが 2^n 単位で分割されて

いるので、簡単な演算(割算)で求めることができる。次に、セクションをサーチエリアとして、この中に存在する線分と交差点データをセクションテーブルとセクションデータを参照することによりロードする(ステップS53～S55)。ロードしたデータを基に、現在地からサーチエリア内の全ての線分までの距離(線分に対する垂線の長さ)、全ての交差点までの距離を計算し、それらを比較することによって最近傍線分と最近傍交差点を得ることができる(ステップS56)。サーチを行なう際のスピードは、線分の本数や交差点の個数に比例するが、前述したデータ構造に基づくサーチ方式によれば、サーチエリア(セクション)が小さく、計算の対象となる線分の本数や交差点の個数が少ないので、高速サーチが可能となるのである。

ところで、ナビゲーションシステムにおいては、種々の縮尺の地図データを表示する際、全ての縮尺の地図データを持っていると、表示は簡単にしかも高速に行なえるが、その半面データサイズが

大きくなるというデメリットがある。逆に最も縮尺の大きい地図データだけを持っていてその他の縮尺を単純な縮小によって表わす場合、データサイズは小さくなるが表示が遅くなるという欠点を持つ。

これに対し、本実施例においては、第8図～第10図に示したデータ構造から明らかなように、データサイズを小さくするために最も縮尺の大きい地図データだけを持ち、更に他の縮尺のデータを表示する際は表示用の管理ファイル及び間引きデータを用いることによって表示の高速化を図っている。以下、第27図のフローチャートに従って、CPU7によって実行される地図の拡大・縮小の手順を説明する。

CPU7は先ず、表示すべき縮尺が入力装置9からキー入力されたことを判別すると(ステップS60)、現在地($Crntx, Crnty$)から縮尺に対応したエリアNo. (Anx, Any)を求め(ステップS61～S63)、続いてその縮尺のピクチャーIDを参照し(ステップS64～S66)、

先頭アドレスとデータサイズによって地図データをロードしてV・RAM上の16個のエリアにそれぞれ描画する(ステップS67)。このように、表示管理用のピクチャーIDによって、表示すべき識別された地図データの参照が(縮尺が小さくなるに従って表示する道路、地名等を垂直行なものにする)ができるので、表示の高速化が実現できるのである。

また、ポリゴンとラインデータに対しては、第11図及び第12図で説明したように、表示を省略しても差し支えない点の間引きビットにはその旨の情報が入れているので、5万分の1や10万分の1の地図の描画の際に、この間引きビットをチェックし(ステップS68)、間引きの対象となっている点を除いて描画する(ステップS69)。このように、地図の縮小の際、ディスプレイに表示した場合に、見た目上省略しても差し支えない点の間引いて表示を行なうことにより、表示する線分の数を減らすことができるので、表示のより高速化が図れるのである。

すると、直進しているところで曲がったと誤認して、交差点でもないのに交差点引込みを行なってしまう、現在地が正しい位置からずれてしまうことになる。

そこで、本実施例においては、曲がったことを判断するのに、曲率半径と車速を判断基準に入れることにより、正確な右折・左折の判断を可能としている。以下、CPU7によって実行される右折・左折の判断方法の手順について、第28図のフローチャートに従って説明する。CPU7は先ず、ある一定距離(例えば、15[m])を走行した際に一定角(例えば、40度)以上曲がったときをカーブ(右折又は左折)したと判断する(ステップS70)。しかし、カーブしたときにそのときの曲率半径Rが判断基準最小回転半径である一定値 R_{min} (例えば、3.5[m])以下のときは、そのデータは間違っていると判断し、カーブしたとは判定しない(ステップS71)。これは、自動車の最小回転半径以下では曲がれないからである。更に、車速Sがある判断基準最高

なお、上記実施例では、ポリゴンとラインデータの間引きビットを設け、表示を省略しても差し支えない点の間引きビットにはその旨の情報が入れられるようにしたが、ポリゴンとラインのデータを等間隔でプロットしておき、表示の際に所定の規則(例えば、縮尺5万分の1の場合1つ飛び、10万分の1の場合4つ飛び等)に従って間引くようにしても良く、同様の効果が得られる。

次に、第22図のフローチャートにおけるステップS24のカーブ(右折・左折)の判断方法について説明する。

基本的には、方位センサである例えば地磁気センサ1の出力データに基づいて右折・左折を判別し、曲がったことを検出した場合に、ステップS28以降の処理によって交差点引込みを行なうのである。しかしながら、地磁気センサ1は外乱に弱く、路切通過時、鉄橋通過時、自転車の側を大きな車(例えば、トラック、バス)が通過した際、その出力データに大きな誤差が含まれることになる。このデータをそのまま右折・左折判断に利用

速度である一定速度 S_{max} (例えば、40[km/h])以上の場合は、交差点を曲がることは通常は考えられないので、この速度以上では、カーブしたとは判定しない(ステップS72)。また、右折・左折の判定は、例えば、東を方位0度、北を方位90度、西を方位180度、南を方位270度とすると、その方位の増減によって行なうことができる(ステップS73)。すなわち、方位が増える方向が左折(ステップS74)、方位が減る方向が右折(ステップS75)となるので、これにより右折・左折を判断できるのである。

なお、曲率半径Rは、第29図に示すように、ある点aにおける車両の方位とその点aから一定距離 l だけ走行した点bにおける車両の方位とのなす角度を θ [ラジアン]とすると、 $l = R \cdot \theta$ であるから、この式を変形して得られる次式

$$R = l / \theta$$

から求めることができる。

また、第22図のフローに於いた処理によって行なわれる交差点引込み等により、現在地がディ

スプレィ上に表示されている地図の道路上に常に位置するように制御されるが、例えば交差点間の距離が長い場合には、その間現在地の微小修正が行なわれるのであるが、センサの精度、計算誤差、地図精度等による距離誤差により、前回引込んだ交差点からの実際の現在地と地図上の現在地とに距離差が生じ、その誤差は交差点間の距離が長い程大きくなる。このような場合、次に引込みを行なうべき交差点の近傍に複数の交差点が近接して複数あると、間違った交差点に引込みを行なう可能性がある。そこで、本実施例では、交差点間において一定距離だけ走行したら、いわゆる走行距離による引込みを行なうようにしている。以下、その手順を第30図のフローチャートに従って説明する。

まず初期値を設定する(ステップS80)。この初期値としては、ある確定した現在地が必要となるが、これは使用者が最初に設定するか、交差点など確定した点へ引き込んだ場合の現在地を利用できるし、またすでに確定した現在地ならば不

揮覚性メモリにそのデータを登録しておけば、一度だけセットすれば良いことになる。この確定した現在地で走行距離をゼロリセットし(ステップS81)、交差点を曲がったか(ステップS82)、一定距離を走ったか(ステップS83)を常に監視しながら、一定距離走ったときに、地図データに基づいてゼロリセットした地図上の点(前回検出位置)からこの一定距離の点を求めてその点に現在地を変更し引込みを行なう(ステップS84)。一定距離を走る間は、見掛け上一番近い線分に垂線をおろし、その交点に引込みを行なうことにより(ステップS85)、ディスプレイ上に表示された地図の道路上に車両の現在地をのせることができる。車両が曲がったことを検出した場合には、交差点引込みを行なう(ステップS86)。この交差点引込みは先述した通りである。

なお、交差点で曲がったという判断にも、この走行距離による引込みが有効に使える。すなわち、交差点間の距離と走行距離により曲がった交差点を地図データより判断できるのである。

発明の効果

以上説明したように、本発明によれば、地図の道路上の各位置を数値化して地図データとして記憶しておき、走行距離センサの出力に基づいて一定距離だけ走行したことを検出する毎に、地図データに基づいて道路上の前回検出位置から前記一定距離だけ離れた道路上の位置を検出し、この検出位置を現在地とすることにより、例えば引込みが行なわれる交差点間の距離が長くても、センサ精度、計算誤差、地図精度等に起因する距離誤差を一定距離毎に補正できるので、車両の現在地を常に正確に認識することができることになる。

—以下余白—

4. 図面の簡単な説明

第1図は本発明に係る車載ナビゲーション装置の構成を示すブロック図、第2図(A)～(C)乃至第13図(A)、(B)は第1図における記録媒体に記憶される地図情報のデータ構造を示す図、第14図(A)～(C)はV-RAM上の画面構成を示す図、第15図乃至第18図は第1図におけるCPUによって実行される基本的な手順を示すフローチャート、第19図乃至第21図は方位補正係数 θ_s の求め方を示す図、第22図はCPUによって実行される交差点引込みルーチン及びパターン引込みルーチンの手順を示すフローチャート、第23図及び第24図は地図上の現在地と最近傍線分との位置関係を示す図、第25図は道路への引込みを行なう他の方法を示す図、第26図は最近傍線分及び交差点をサーチする手順を示すフローチャート、第27図は地図の拡大・縮小の手順を示すフローチャート、第28図は右折・左折の判定方法の手順を示すフローチャート、第29図は曲率半径の求め方を示す図、第30図

は走行距離による引込み方法の手順を示すフロー
チャートである。

主要部分の符号の説明

- 1 …… 地磁気センサ 2 …… 角速度センサ
5 …… システムコントローラ
7 …… CPU 10 …… 記録媒体
12 …… ディスプレイ 14 …… 入力装置

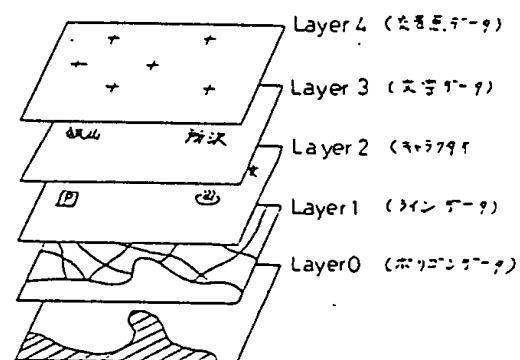
第 3 図

(A)

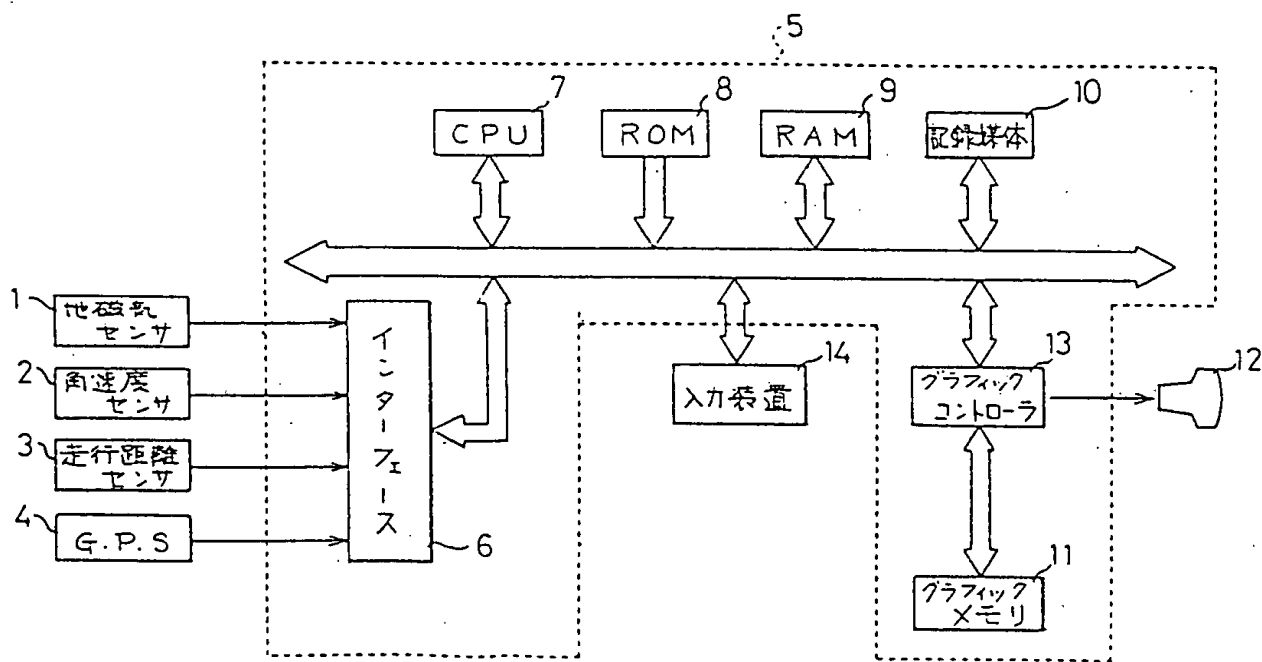
1	地磁気センサ	2	角速度センサ	3	走行距離センサ	4	G.P.S.	5	システムコントローラ	6	インターフェース	7	CPU	8	ROM	9	RAM	10	記録媒体	11	グラフィックメモリ	12	ディスプレイ	13	グラフィックコントローラ	14	入力装置
---	--------	---	--------	---	---------	---	--------	---	------------	---	----------	---	-----	---	-----	---	-----	----	------	----	-----------	----	--------	----	--------------	----	------

出 願 人 バイオニア株式会社
代 理 人 弁 理 士 松 村 元 彦

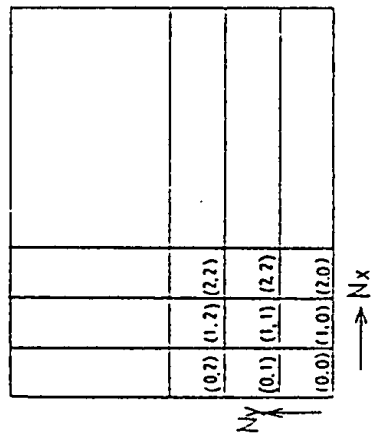
(B)



第 1 図



第 4 図 (A)



(B)

$$\begin{cases} N_x = \{x \text{ (crnt } x)\} \\ N_y = \{y \text{ (crnt } y)\} \end{cases}$$

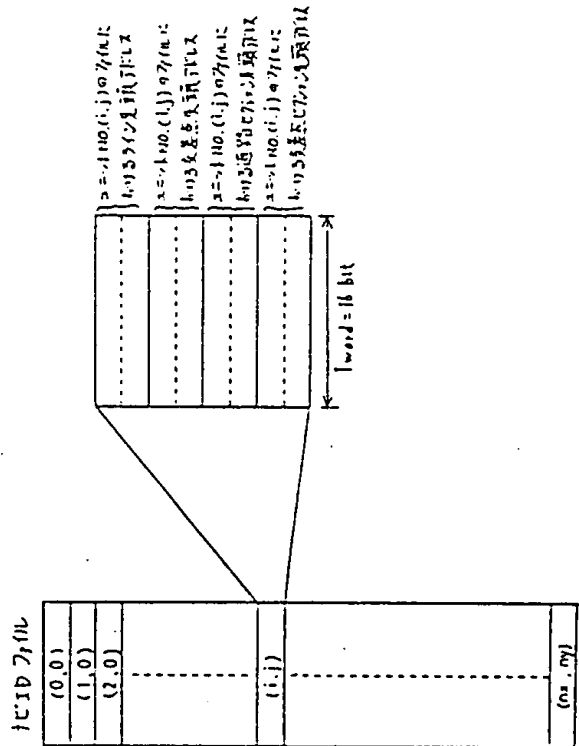
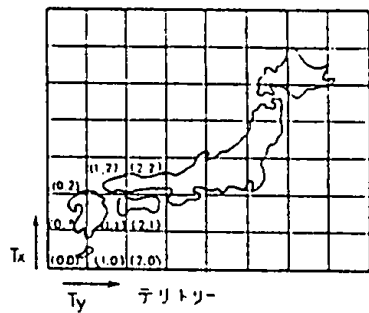
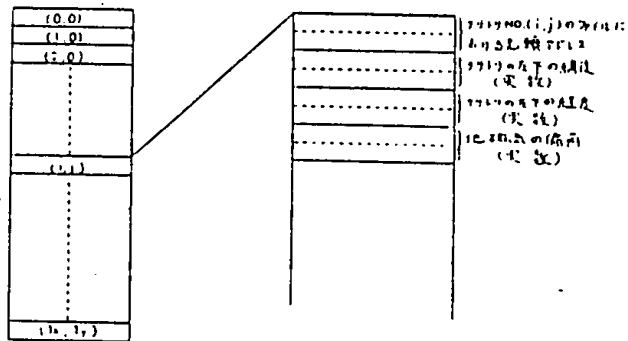


図 2 (A)



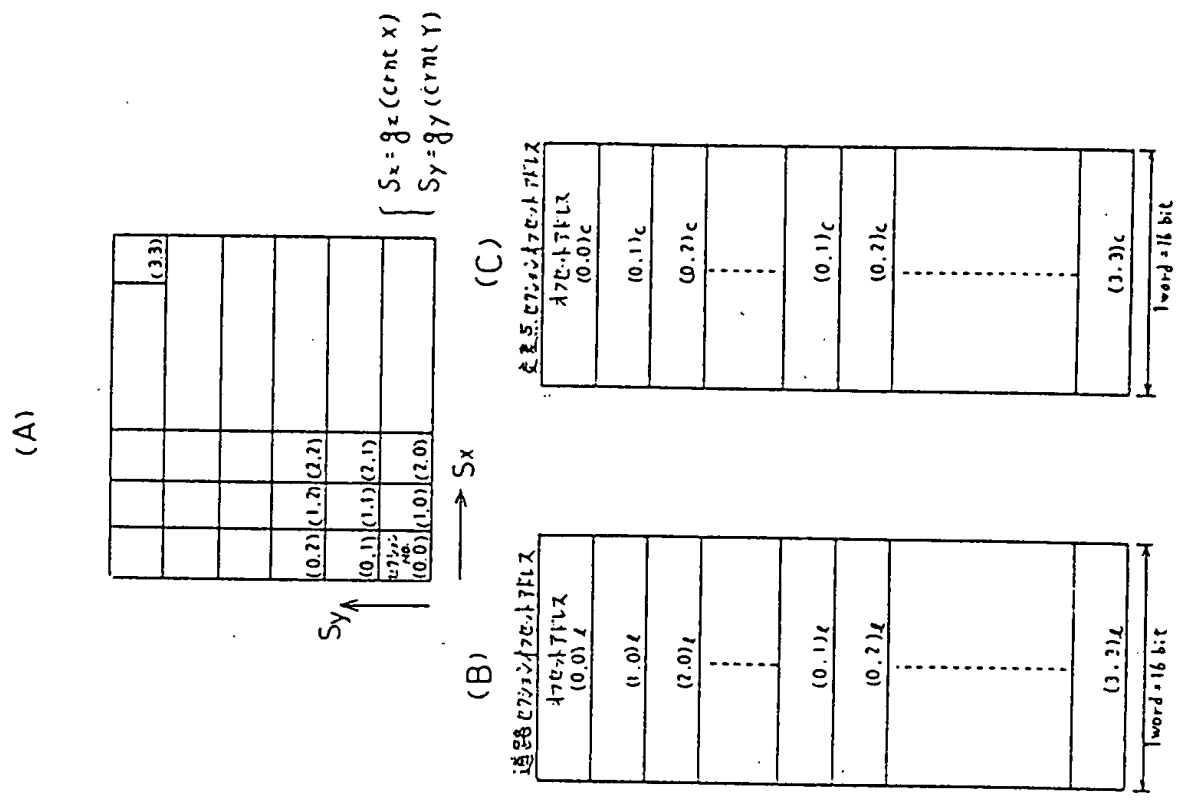
(C)



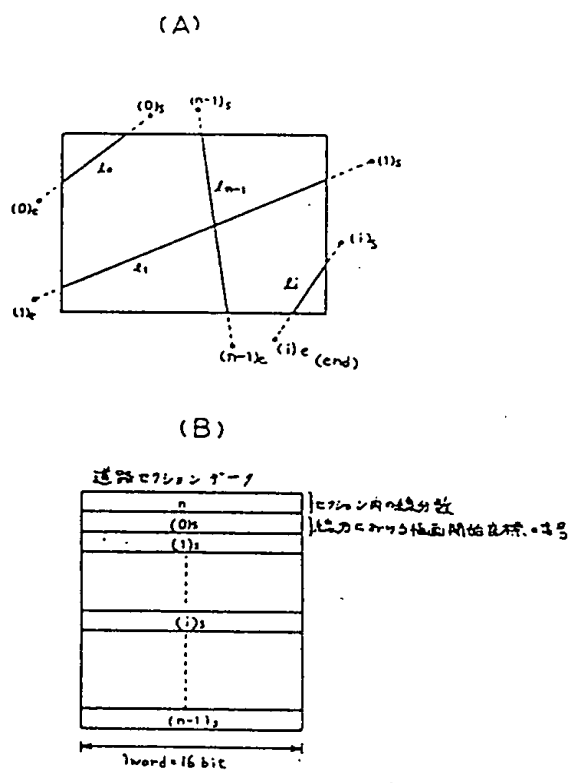
(B)

テリトリ ID	テリトリ (0,0)	テリトリ (1,0)	...	テリトリ (i,j)	...	テリトリ (k,Ty)
------------	---------------	---------------	-----	---------------	-----	----------------

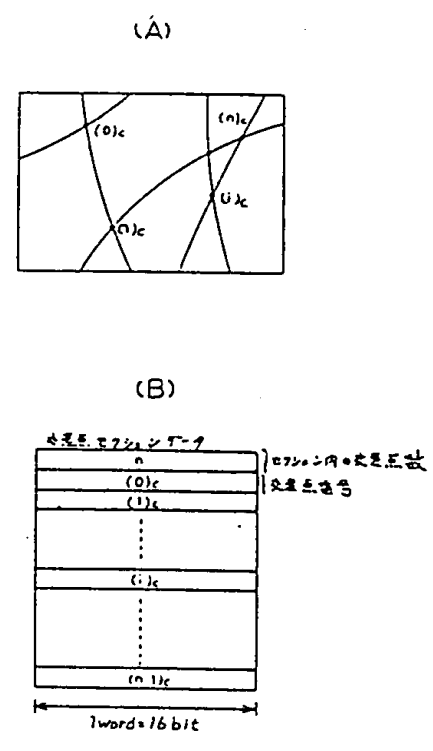
第 5 図



第 6 図



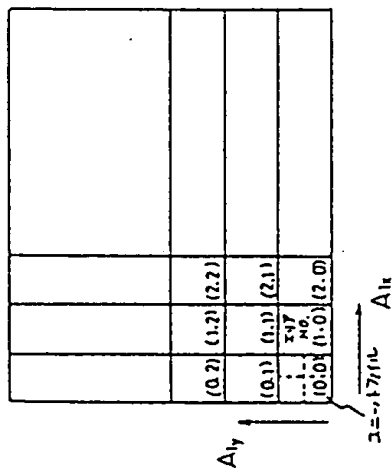
第 7 図



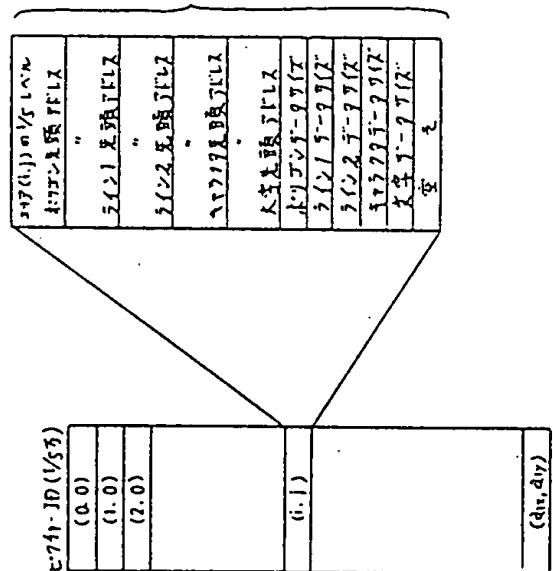
第 9 図

(A)

1/5 区画



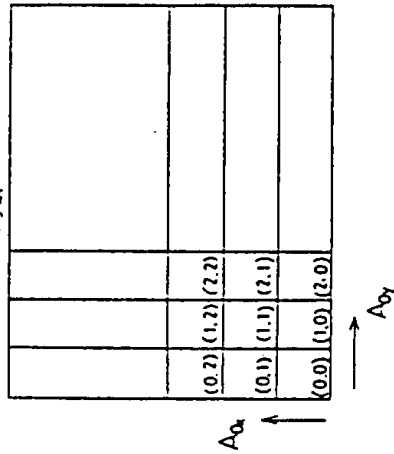
(B)



各 1/5 区画 4 個分

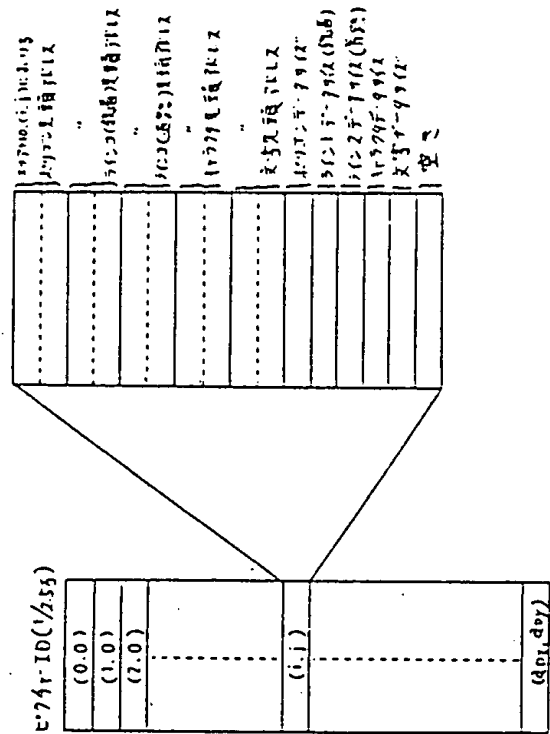
第 8 図 (A)

1/2.5 区画



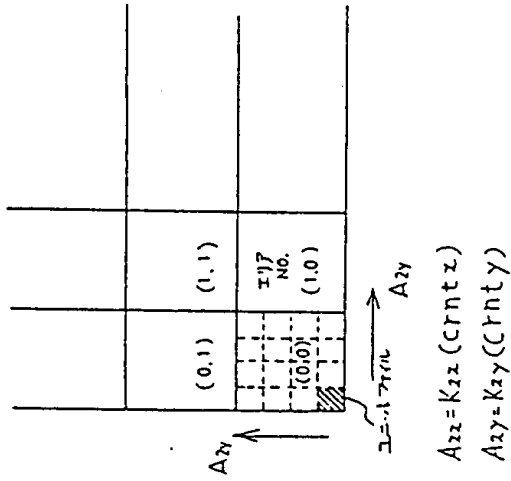
$\begin{cases} A = K_{0x} (crnt\ x) \\ A = K_{0y} (crnt\ y) \end{cases}$

(B)

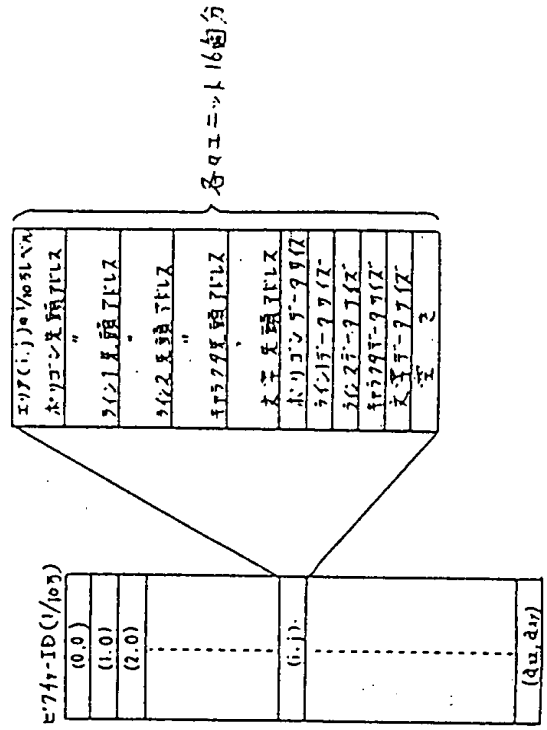


第 10 図

(A)

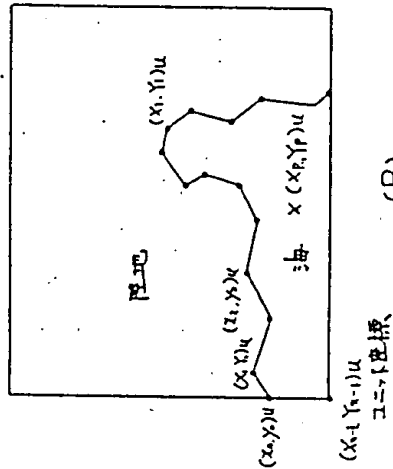


(B)

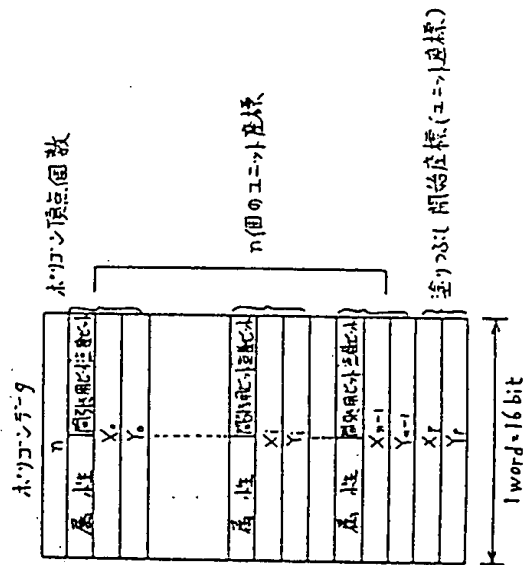


第 11 図

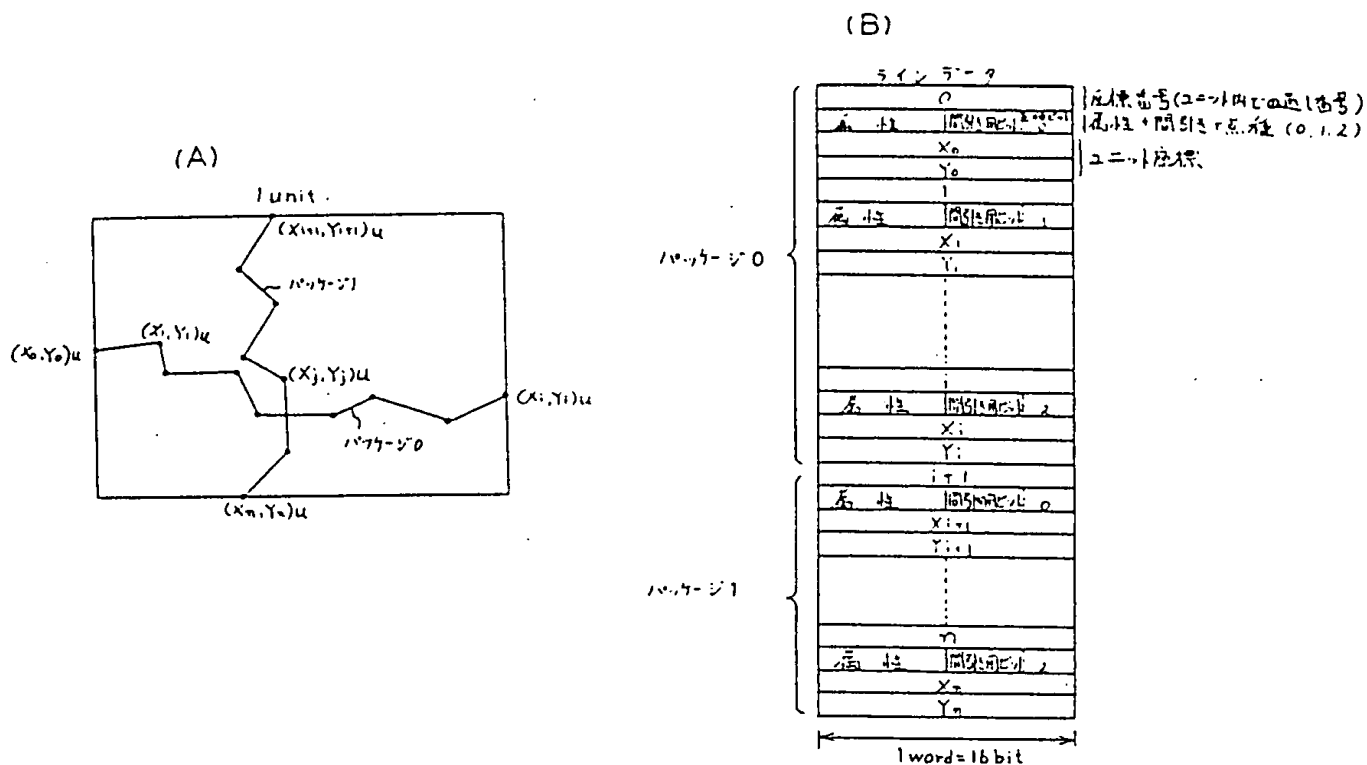
(A)



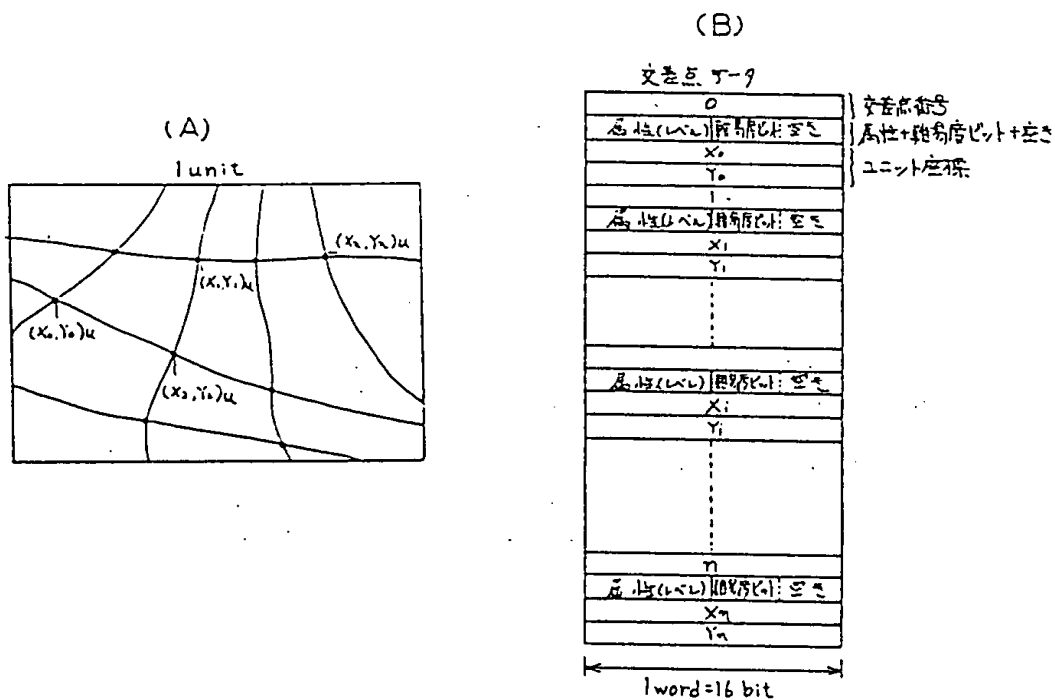
(B)



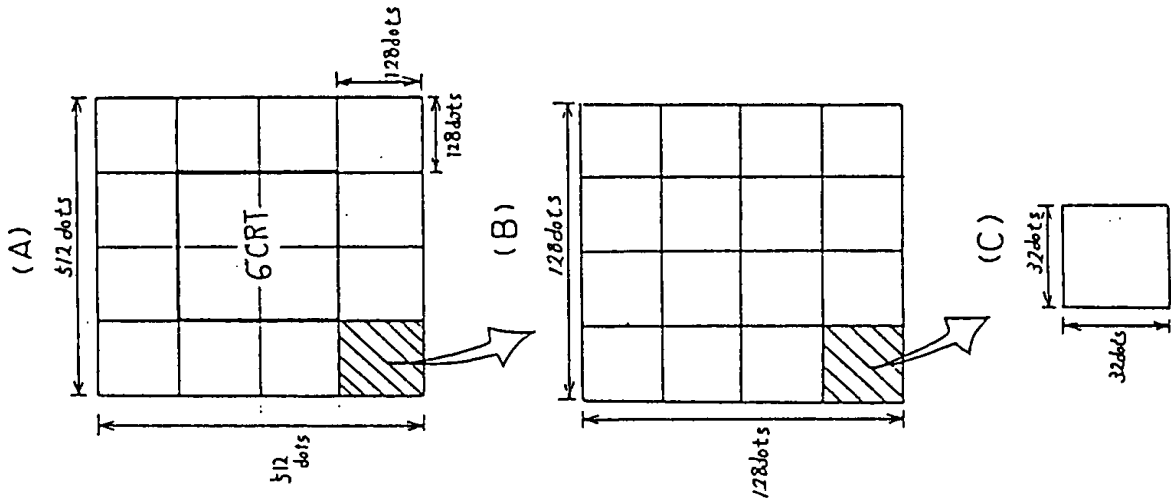
第 12 図



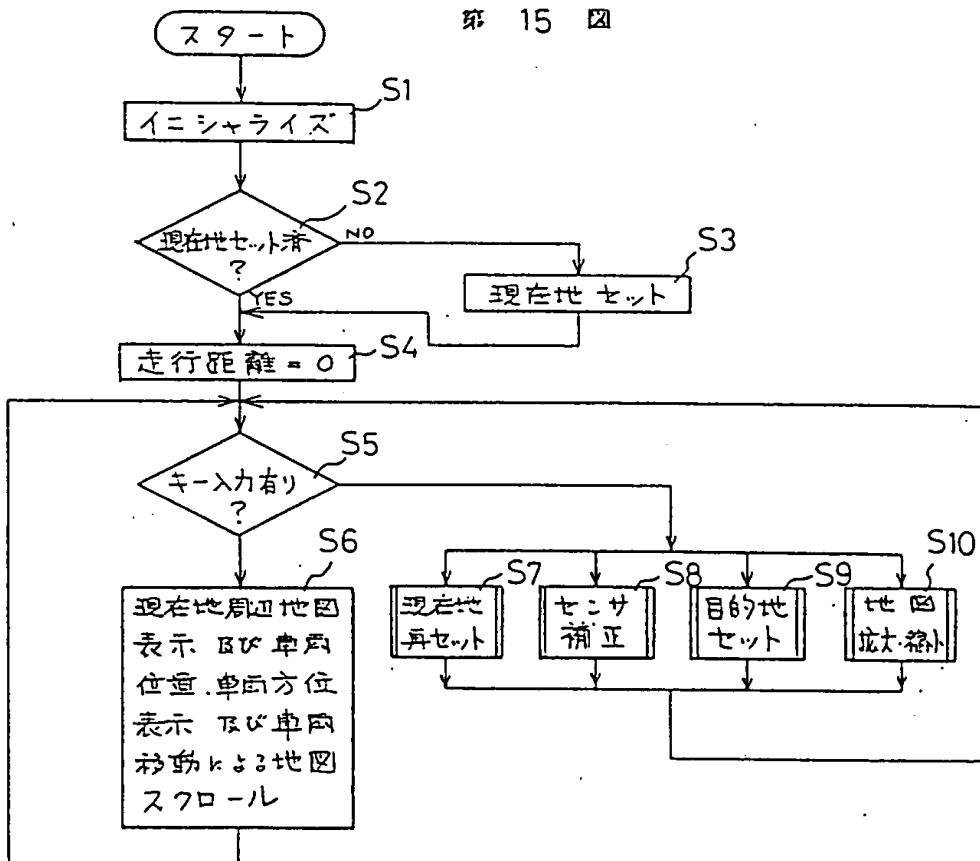
第 13 図



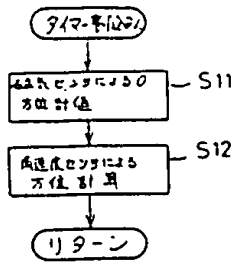
第 14 図



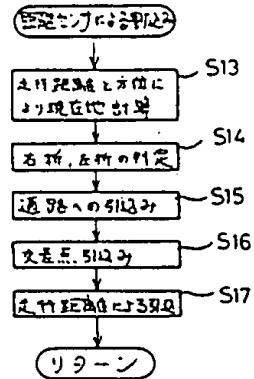
第 15 図



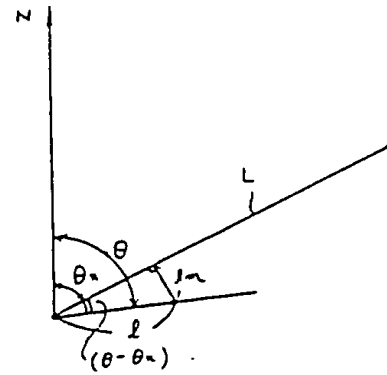
第 16 図



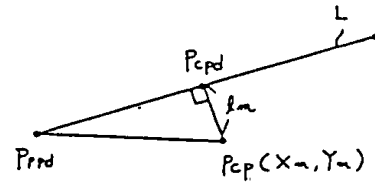
第 17 図



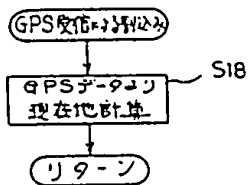
第 23 図



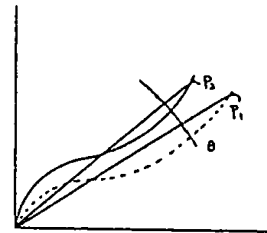
第 24 図



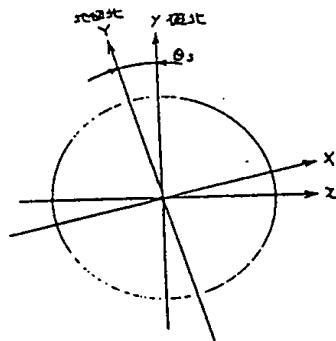
第 18 図



第 19 図

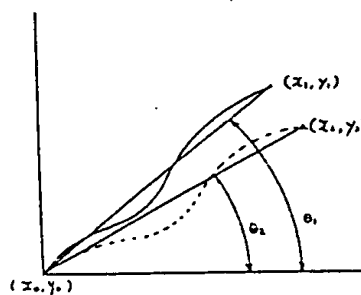


第 20 図



$$\begin{cases} X = x \cos \theta_0 + y \sin \theta_0 \\ Y = -x \sin \theta_0 + y \cos \theta_0 \end{cases}$$

第 21 図



$$\begin{aligned} \theta_1 &= \theta_0 - \theta_2 \\ &= \tan^{-1} \left(\frac{y_1 - y_0}{x_1 - x_0} \right) - \tan^{-1} \left(\frac{y_0 - y_2}{x_0 - x_2} \right) \end{aligned}$$

図 22

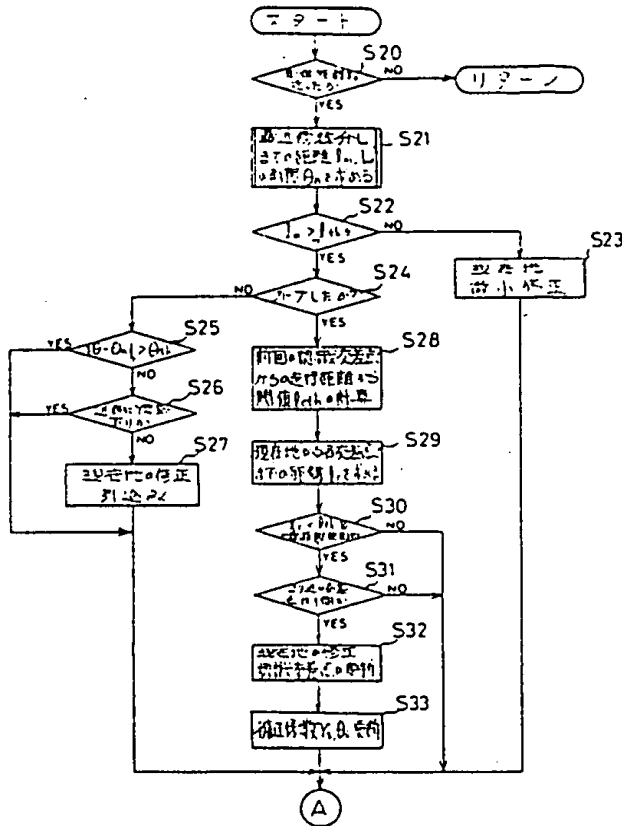


図 25

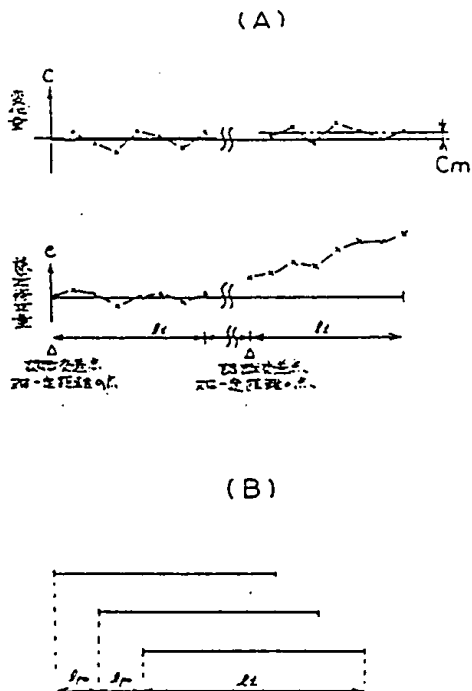
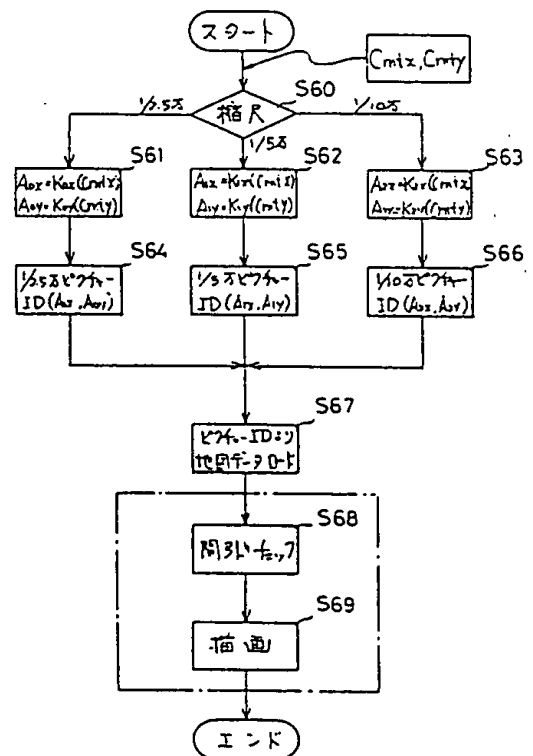
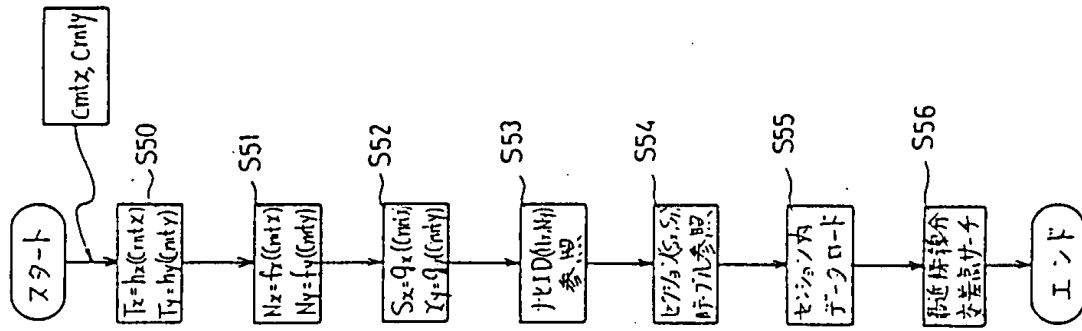


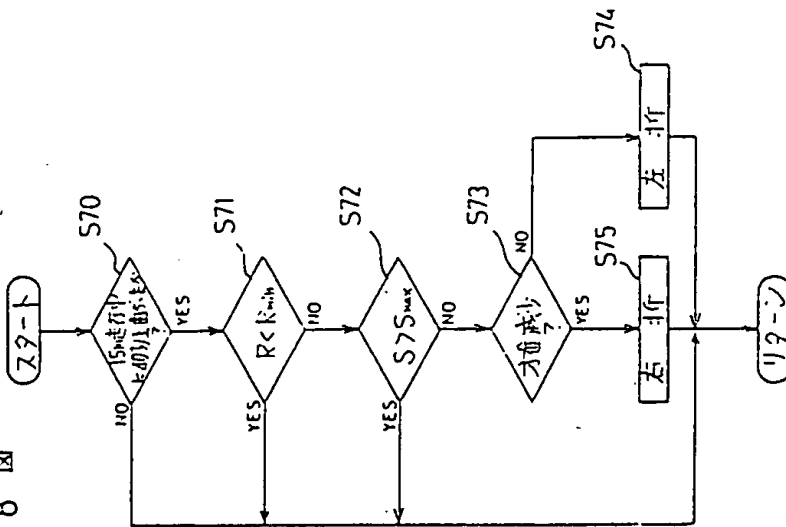
図 27



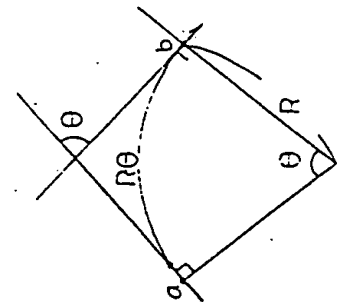
第 26 図



第 28 図



第 29 図



第 30 図

